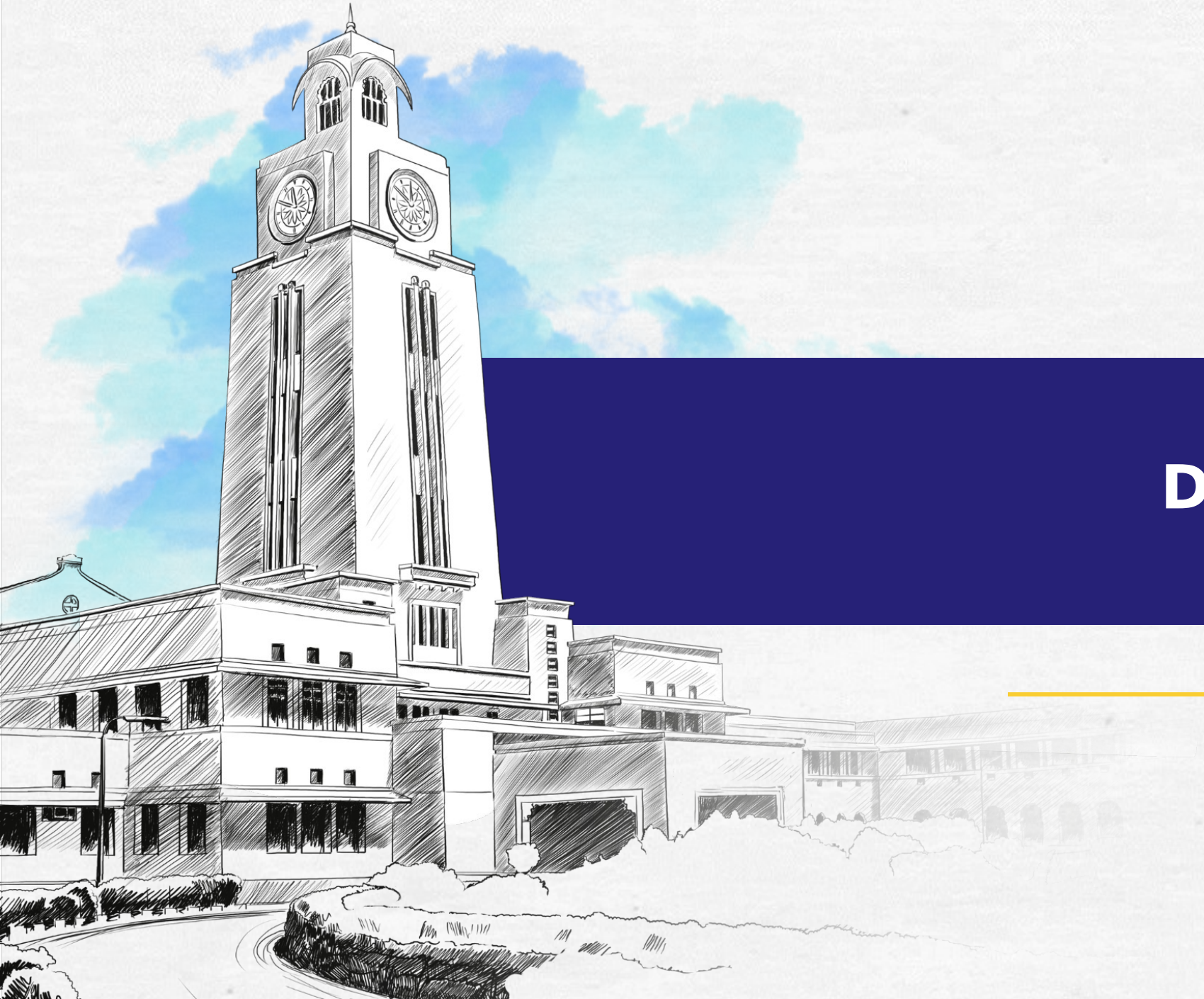


BITS Pilani
Pilani | Dubai | Goa | Hyderabad

Data Architectures

Sunil Bhutada

Agenda

- Introduction
- Categorization
- Generalized Architecture of Big Data Systems
- Lambda & Kappa for Big Data Systems

Introduction

Definitions

TOGAF (The Open Group Architectural Framework)
A description of the structure and interaction of the enterprise's major types and sources of data, logical data assets, physical data assets, and data management resources

DAMA – DMBOK (Data Management Association, DM Body of Knowledge)
Identifying the data needs of the enterprise (regardless of structure) and designing and maintaining the master blueprints to meet those needs. Using master blueprints to guide data integration, control data assets, and align data investments with business strategy.

Data Architecture is the design of systems to support the evolving data needs of an enterprise, achieved by flexible and reversible decisions reached through a careful evaluation of trade-offs

Data Engineering architecture is the systems and frameworks that make up the key sections of the data engineering lifecycle.

Why ?

- Reflects current and future state of data systems that supports organization’s long-term data needs and strategy
- Data requirements changes rapidly
  - **New tools and practices** seem to arrive on a near-daily basis
  - Gives rise to the need of understanding good data architecture
- **To understand needs of business and gather requirements for new use cases**
- **To translates requirements to design new ways to capture and serve data**
- Knowing the trade-offs with design patterns, technologies and tools in source systems, ingestion, storage, transformation and serving data
- Successful data engineering is build upon rock-solid data architecture!

Aspects

Data Architecture

Operational

What

- Encompasses the functional requirements of what needs to happen related to people, processes and technology
- What business processes does the data serve?
- How does organization manages data quality?
- Describes what needs to be done

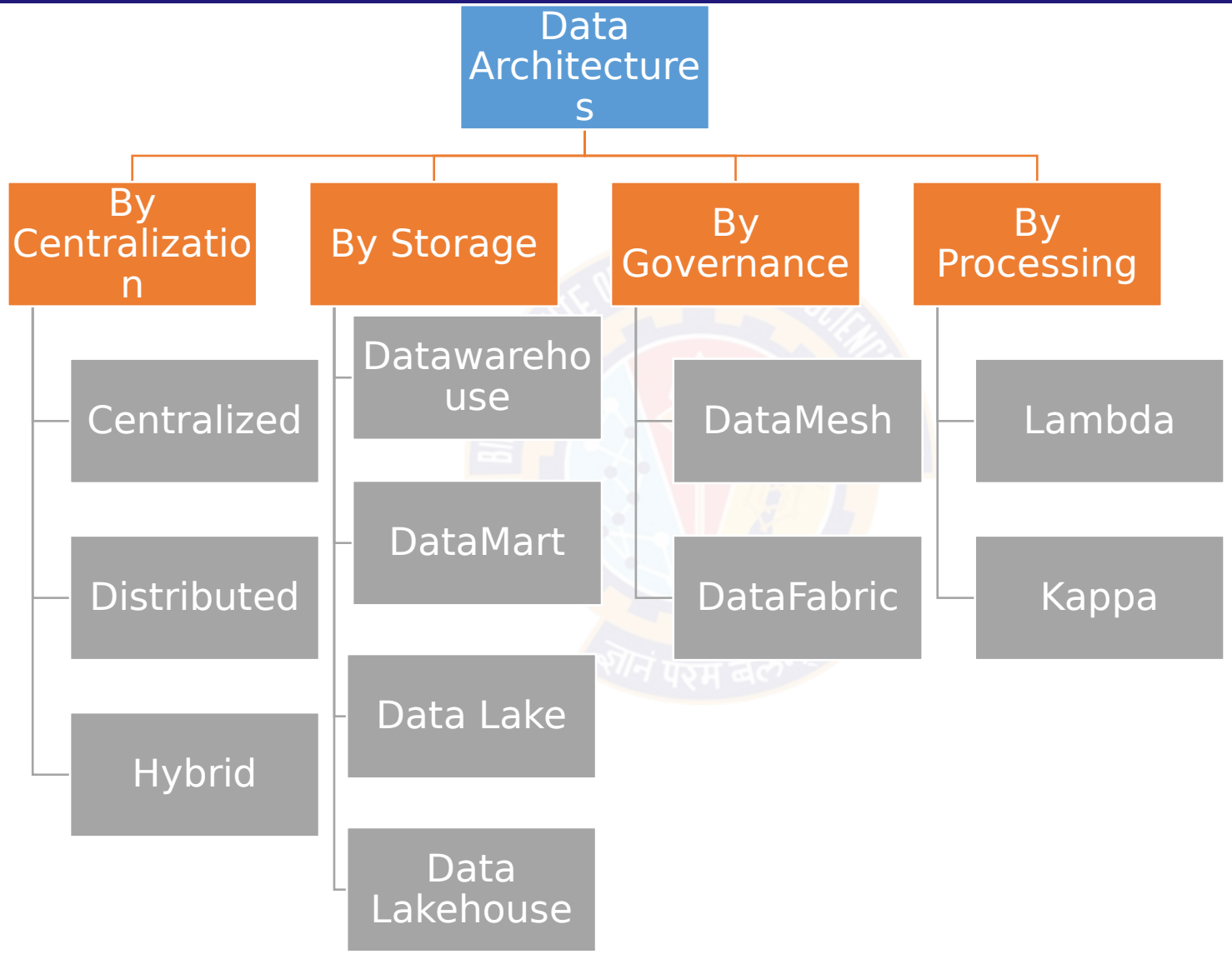
Technical

How

- Outlines how data is ingested, stored, transformed and served along the data engineering life cycle
- How will you move 10 TB of data every hour from source database to data lake?
- Describes how it will happen

Categorization

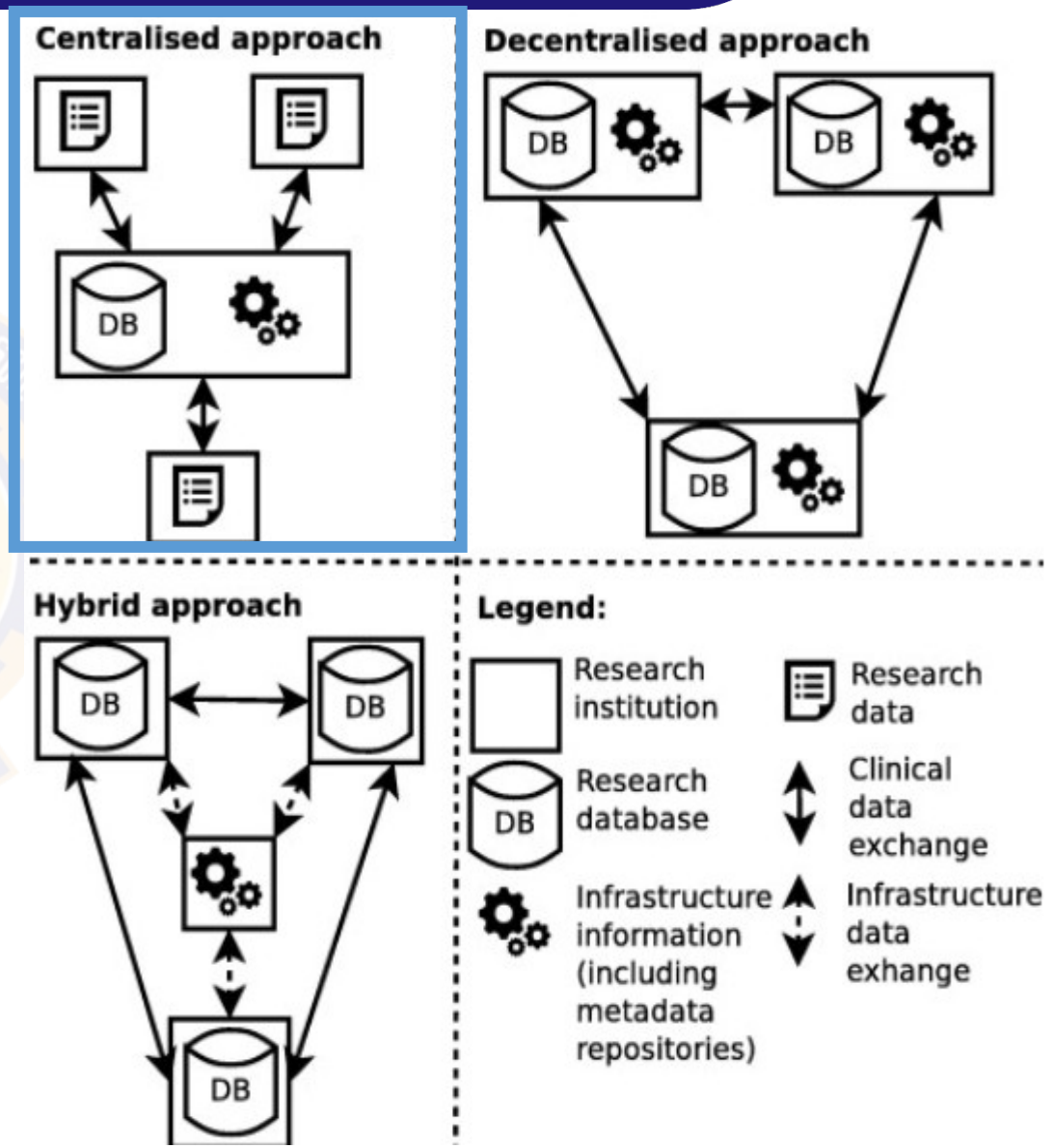
Categorization



Categorization – By Centralization

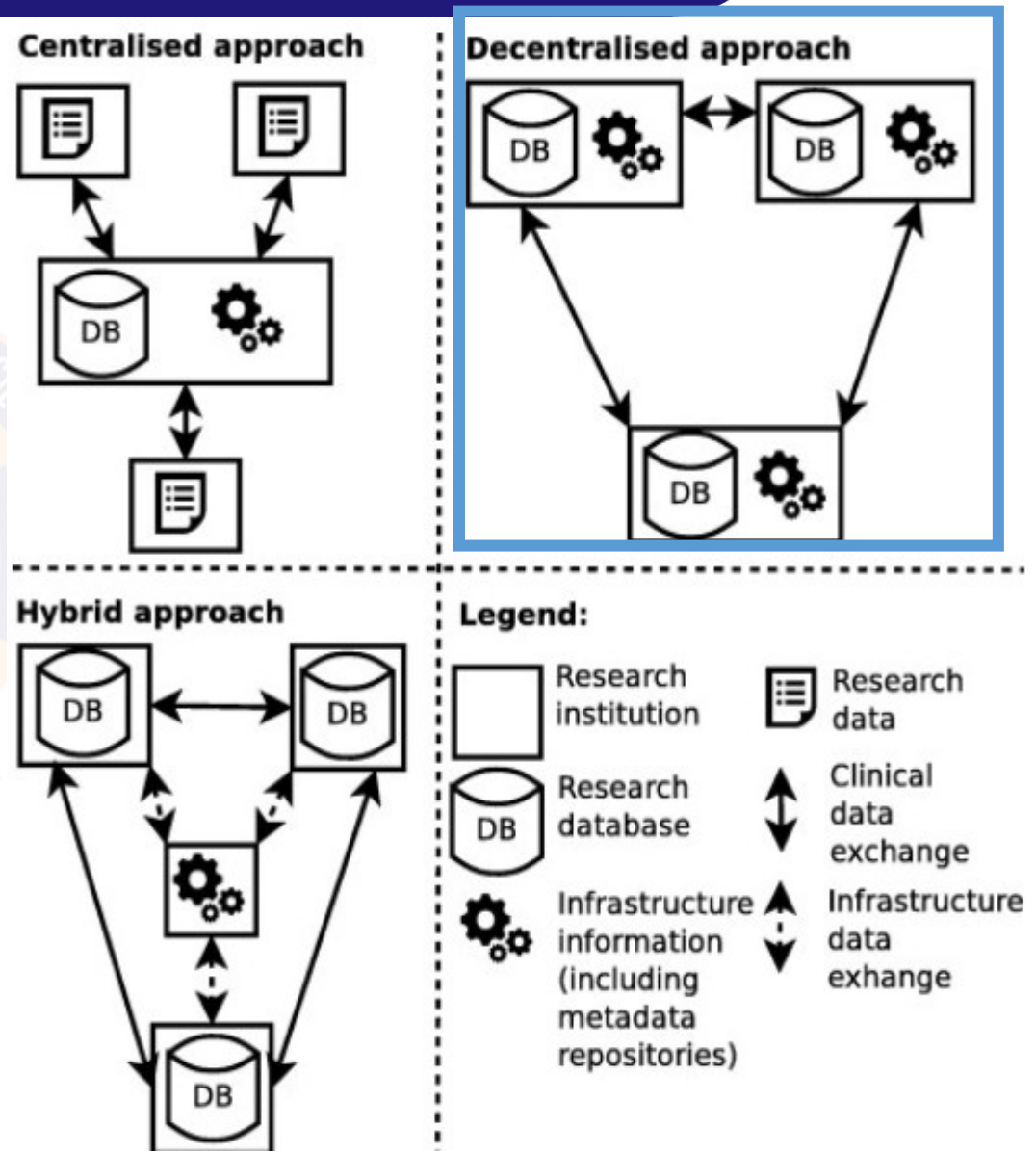
Centralized Data Architecture

- The centralized architecture provides a single point of control for data governance, auditing, and reporting, using authoritative sources, a single intelligent landing area, and a data aggregation layer across business units.
- Simplified enterprise consumption platforms for reporting and analytics are also used.
- Eg: banking or healthcare organizations that operate in highly regulated environments



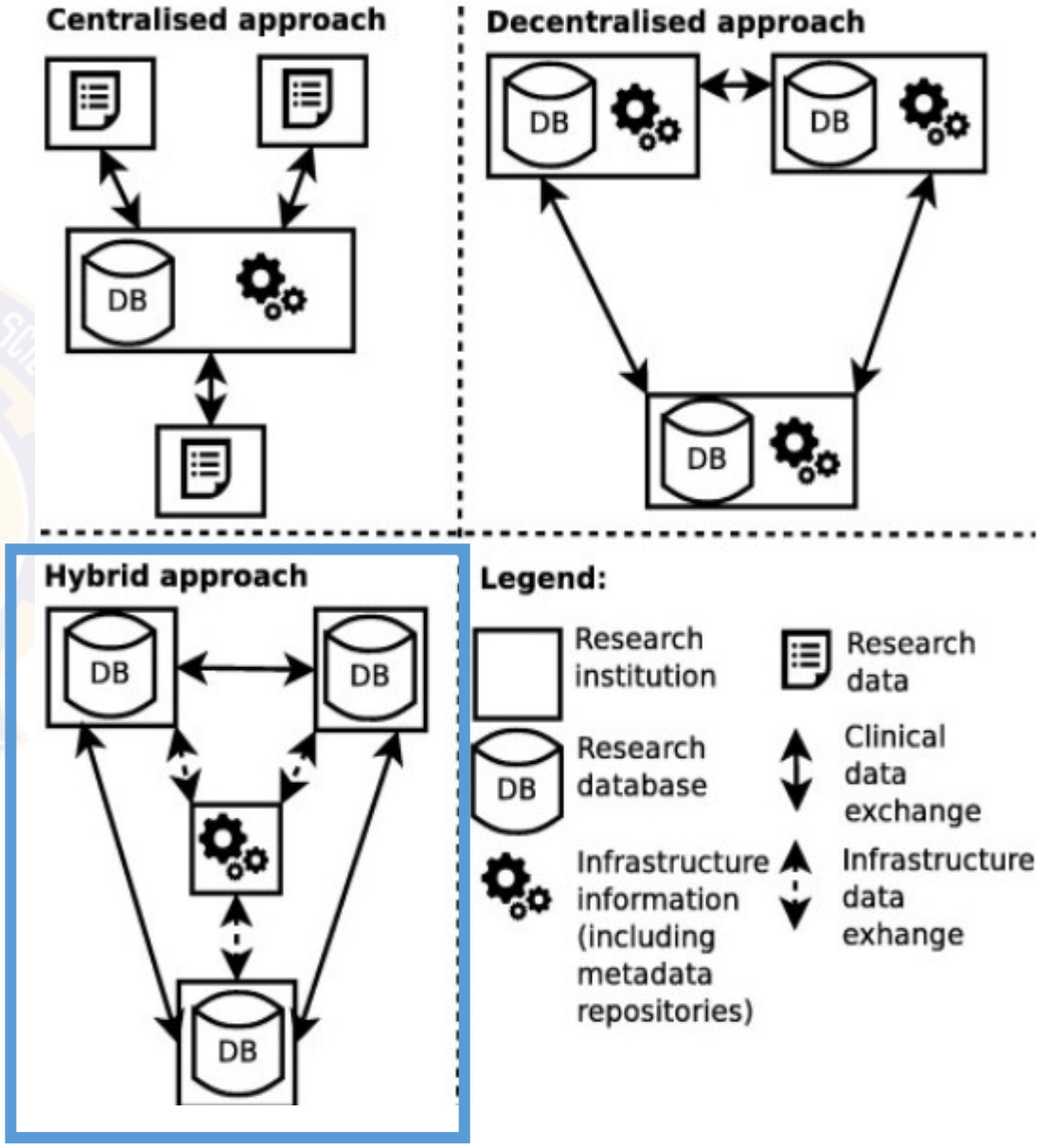
Decentralized Data Architecture

- Data is organized and optimized front to back within business unit silos, each of which includes sources, aggregation, and business unit–level reporting
- Platforms aggregate across business units for enterprise-level reporting
- Eg: Insurance companies, for example, may use this approach, wherein MDM and business domain data are decentralized to accommodate the needs of different customer bases, core systems, and data products.



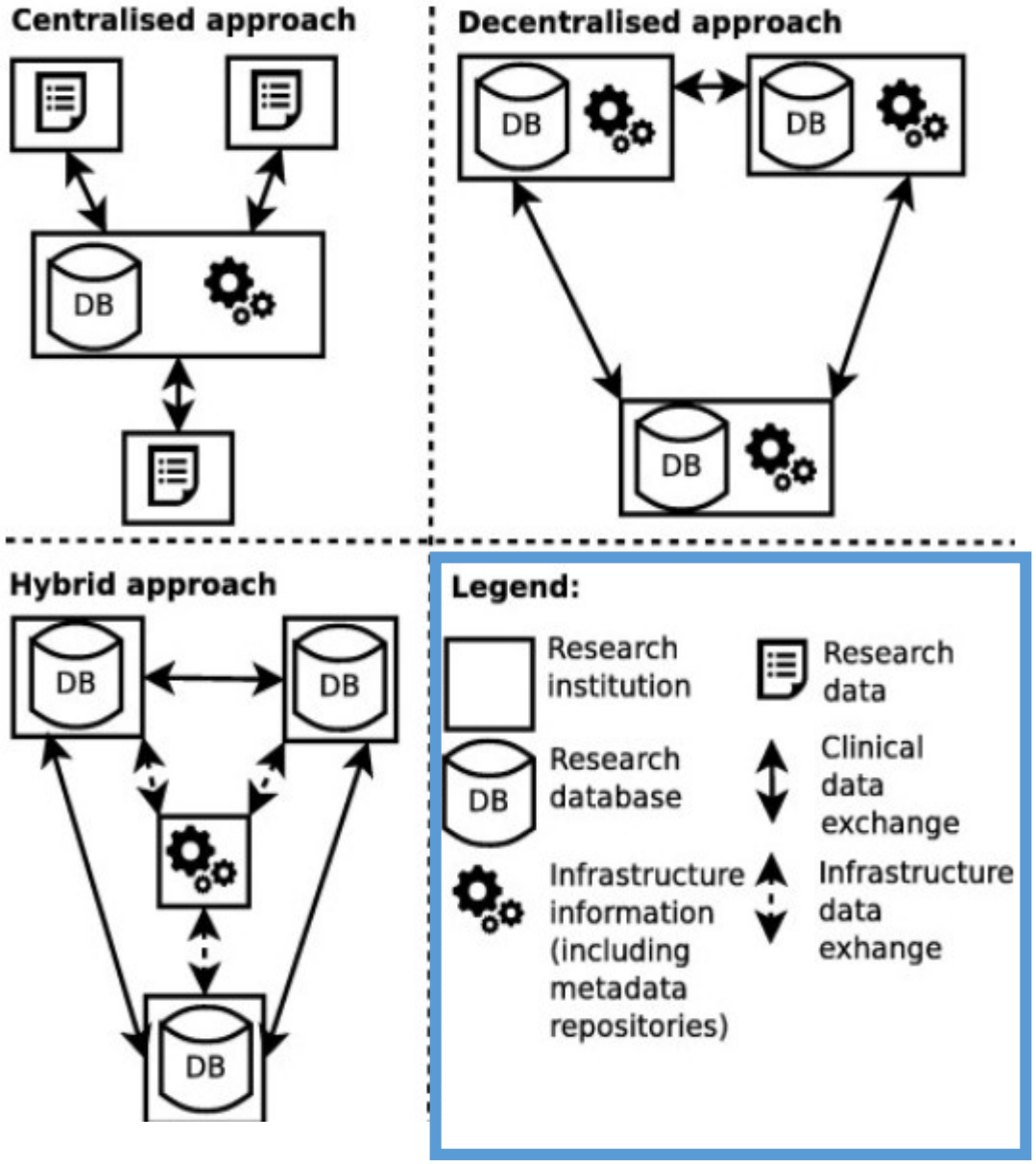
Hybrid Data Architecture

- Within a hybrid data architecture, data and platforms are organized and rationalized by data domain with single golden sources and no duplication across data domains.
- Helpful in operations with rapidly updated data streams and clear alignment of processes within each business unit
- Eg: Telecommunications companies, for example, often use a hybrid architecture with centralized master data management (MDM) and federated data storage for the individual data domains.



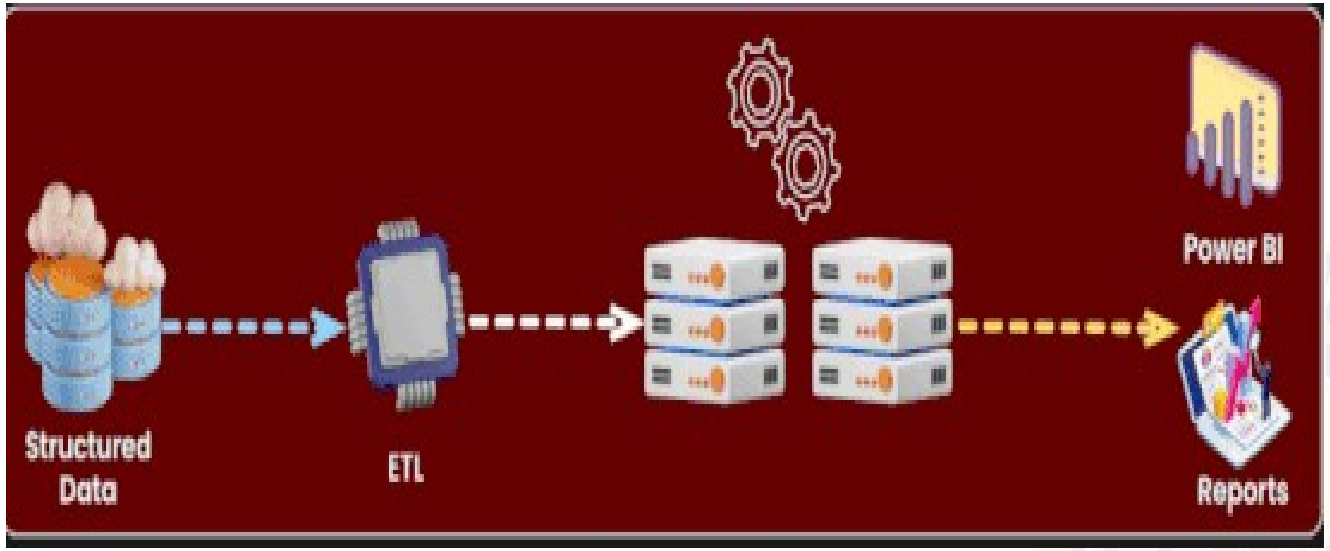
Legend Data Architecture

- The **Legend** in the hybrid data architecture diagram explains the graphical symbols used:-
- **Research institution** – represents the organization or entity involved in research.
- **Clinical Data** – denotes data specific to clinical operations or patient information.
- **Research Data** – indicates data generated from research activities.- Infrastructure & interface (including separate repositories) – shows the technical layer that handles data storage and exchange, including intermediate repositories.
- **Data Exchange** – arrows that depict the flow of data between systems or databases.
- In the **Hybrid approach**, the research database connects through the infrastructure & interface to enable data exchange with the clinical data repository, allowing centralized management (MDM) combined with federated storage for individual domains.



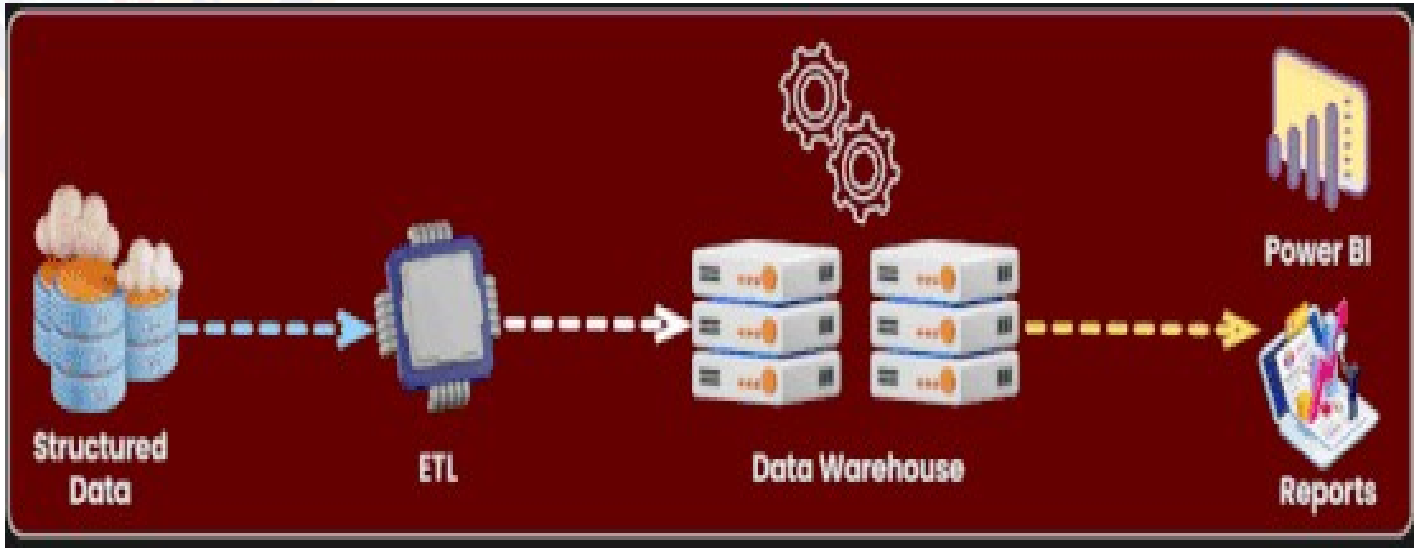
Categorization – By Storage

Data Warehouse



- You are running an ETL process
- The data is already transformed to a defined / structured format
- You want to use it for business analytics
- Which one is the right storage for this?

- Load phase pushes data into DW target database system
- Transformation-on-Read ELT



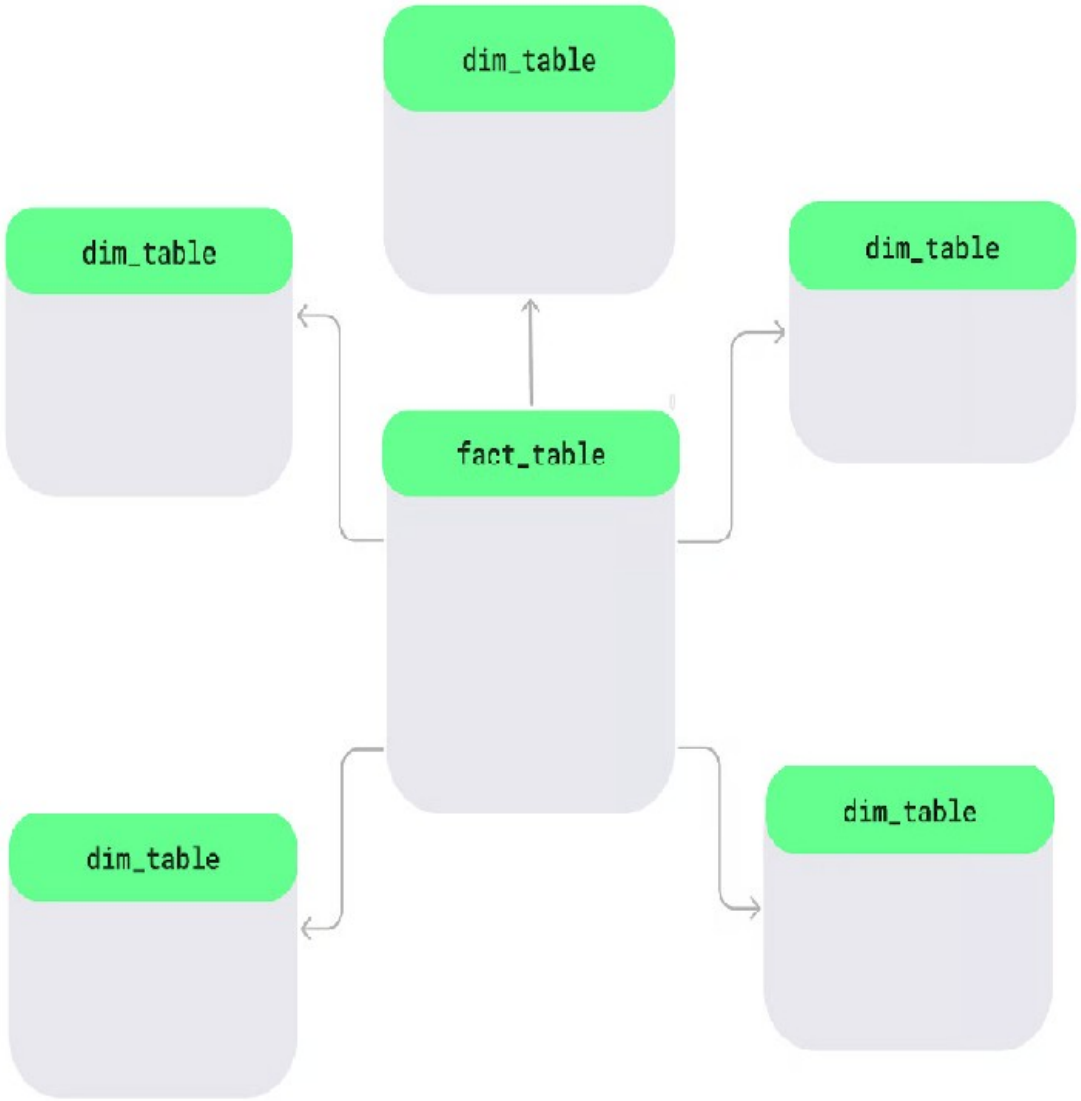
Data Warehouse Schema

- A data warehouse schema is the logical structure or blueprint that organizes how data is stored, related, and queried within a data warehouse.
- It defines tables, keys, and relationships among facts (quantitative data) and dimensions (context data).
- Unlike operational databases (which are optimized for transactions), a data warehouse is optimized for analytics, reporting, and decision support.

A schema ensures data is:

- Organized for fast querying
- Consistent and reusable
- Understandable to business users
- Efficient for aggregation and slicing

Star Schema



- At the center, there's a main table called the fact table, which holds measurable data like sales or revenue.
- Around it are dimension tables, which add details like product names, customer info, or dates.
- This layout forms a star-like shape.

Star Schema - Contd

- **Single-level dimension tables**: The dimension tables connect directly to the fact table without extra layers. Each table focuses on one area, like products, regions, or time, making it simple to use.

- **Denormalized design**: In a star schema, related data is stored together in one table using a denormalized approach. For example, a product table may include the product ID, name, and category in the same place. While this may mean some data repetition, it processes queries faster.

- **Common in data warehousing**: The star schema is used for quick analysis. It can easily filter or calculate totals, so it's likely a good choice for data warehouses where fast insights are required.

- **Advantages:**

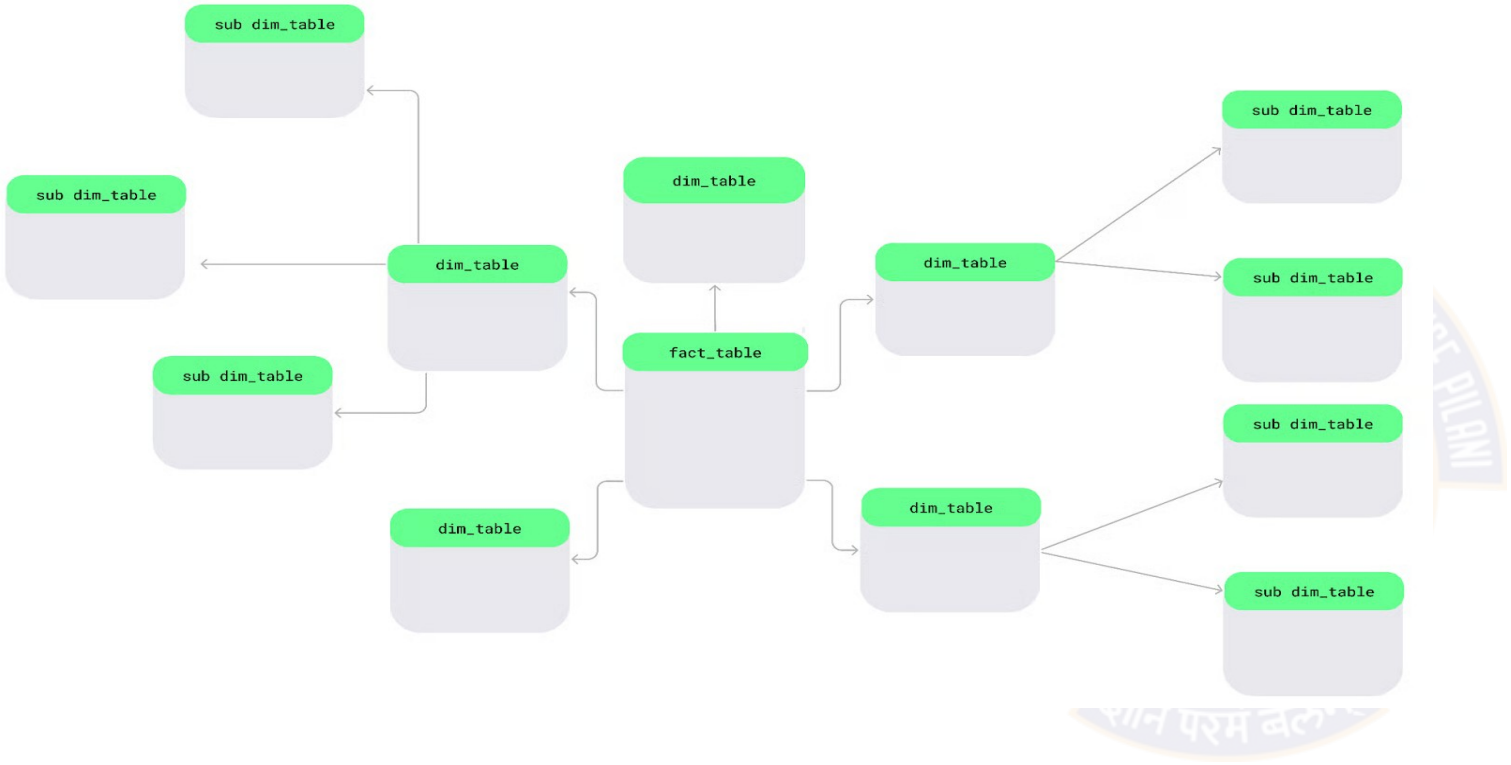
  - **Faster query performance**: The star schema simplifies data retrieval by making queries fast.

  - **Easy to understand**: Its structure is logical and simple to understand, even for non-technical users.

- **Drawback:**

  - due to denormalization, dimension tables often contain repeated information, which **increases**

Snow Flake Schema



- Dimension tables are split into smaller sub-dimensions to keep data more organized and detailed — just like snowflakes in a large lake.

Snow Flake Schema - contd

- **Multi-level dimension tables**: We can break down our dimension tables into smaller, more specific tables.

- **Normalization for storage efficiency**: Unlike star schema, the snowflake schema follows a **normalized design**, which avoids data duplication.

- Suitability for complex data environments: The snowflake schema works best for complex data environments because it uses multi-level tables to handle intricate relationships and hierarchical data structures.

- **Advantages:**
  - **Less data redundancy**: Normalization ensures that the same data is not stored multiple times, which reduces duplication.

  - **Efficient storage for large datasets**: This schema saves storage space by avoiding repeated data, making it ideal for managing large datasets.

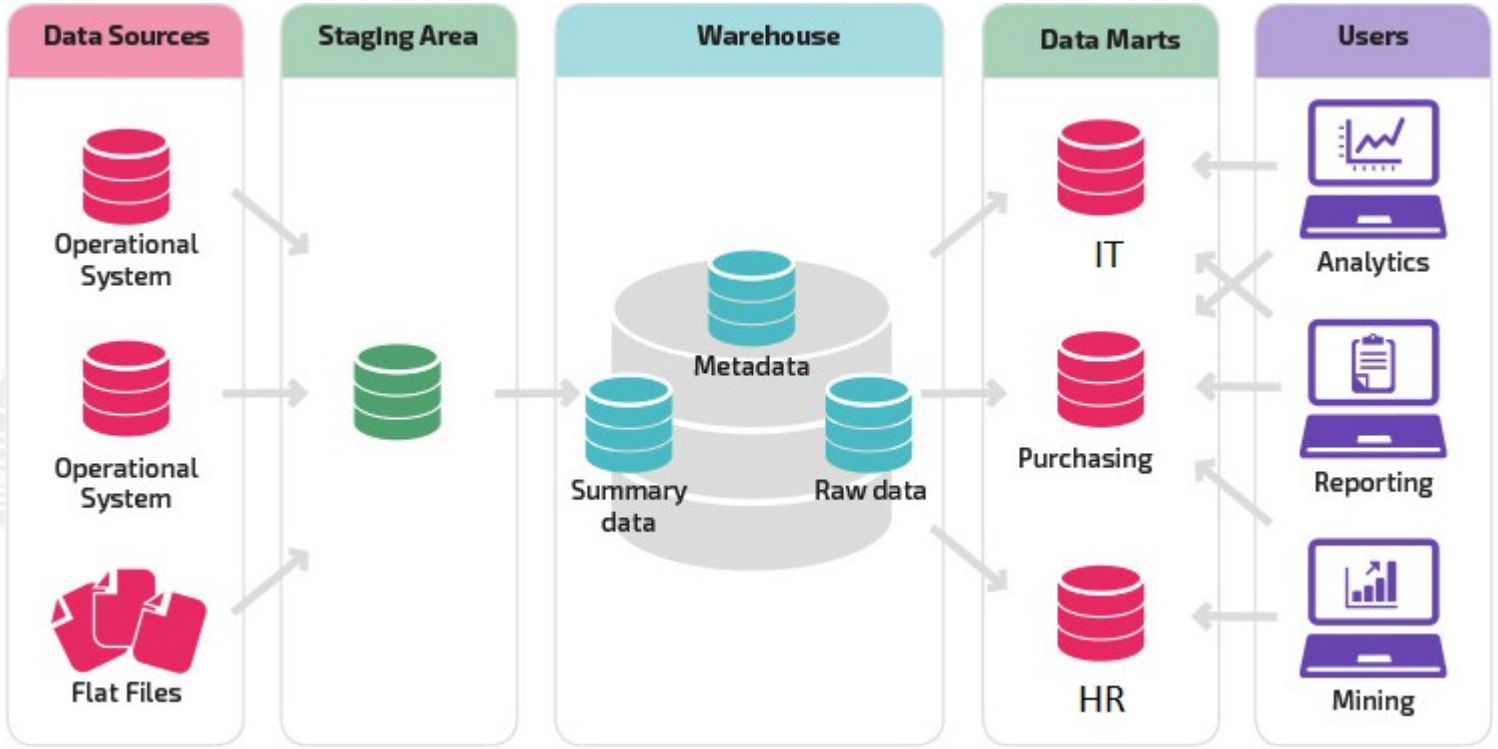
Drawbacks:

- **Queries can be slower** because there are more joins between tables.

- Apart from this, the multi-level structure is more **challenging to design and maintain** than simpler schemas like star schema.

Data Marts

- Consider a scenario where after storing the data in data warehouse, individual departments want the data for their own analytical purpose.
- Eg: HR wants to see how the recruitment is performing, IT team wants to build a dashboard on the frequence of tickets basis the type of ticket.
- So running a query on the entire database will have an impact on the performance and may delay the outcome
- What is the solution?

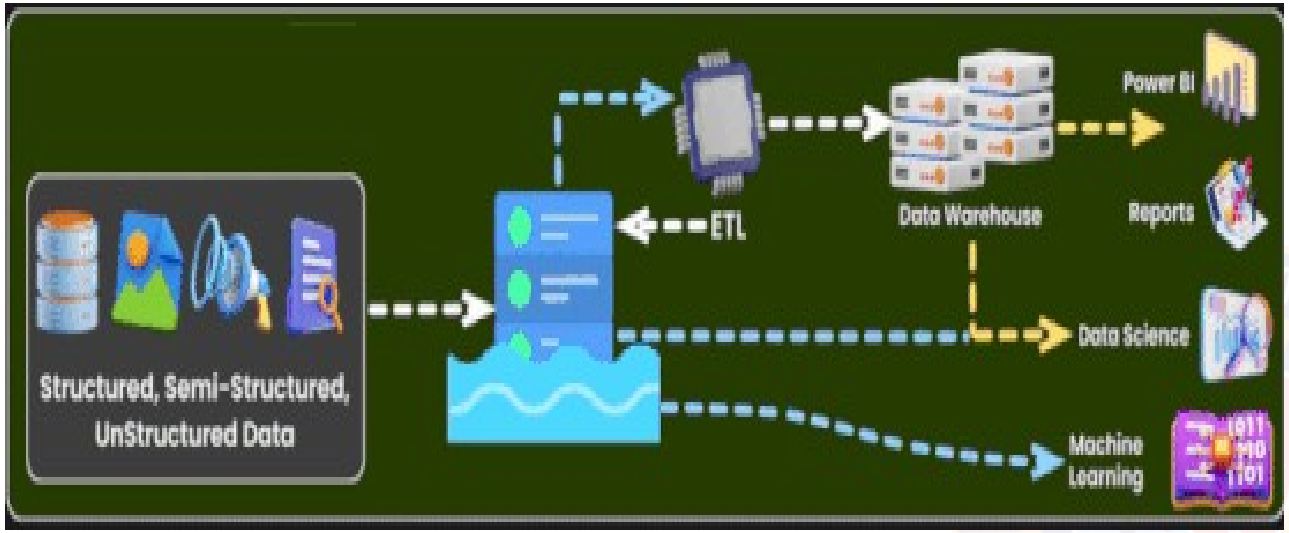


A special case of Data Warehouse

Data Marts

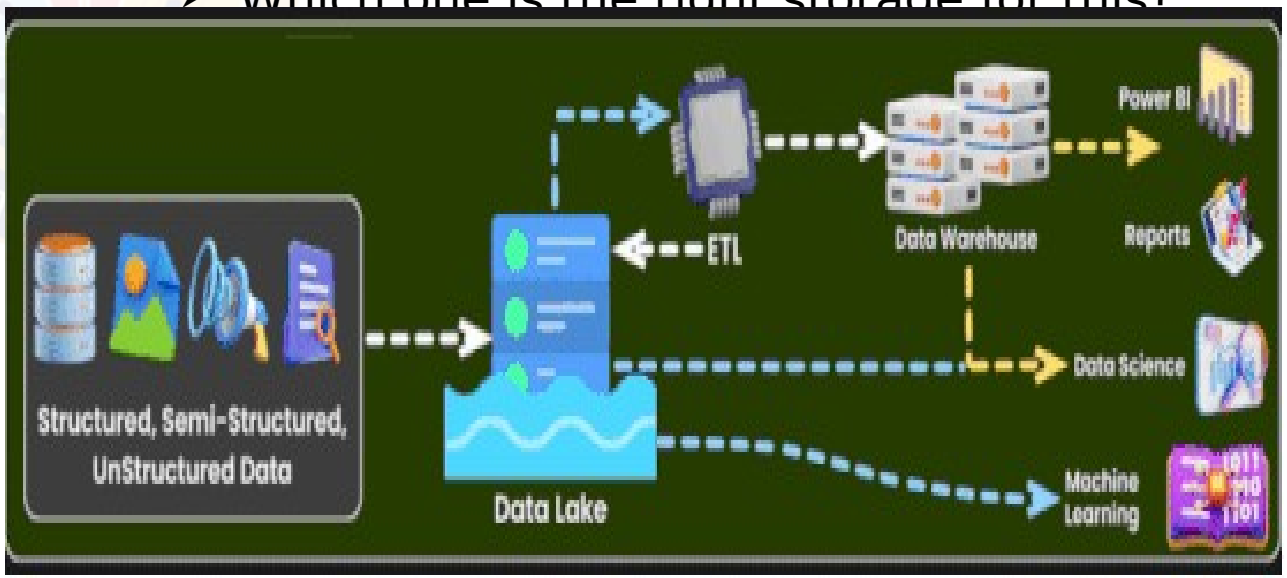
- More refined subset of a warehouse designed to serve analytics and reporting, focused on a single sub organization, department or line of business
- Every department has its own data mart, specific to its needs
- Contrast to full data warehouse that serves the broader organization or business
- Makes data more easily accessible to analysts and report developers
- Provide additional stage of transformation beyond that provided by initial ETL/ELT pipelines
- Significantly improves performance if reports or analytics requires complex joins and aggregations of data, especially when raw data is large
- Transform processes can populate data mart with joined and aggregated data to improve performance of live queries

Data Lake

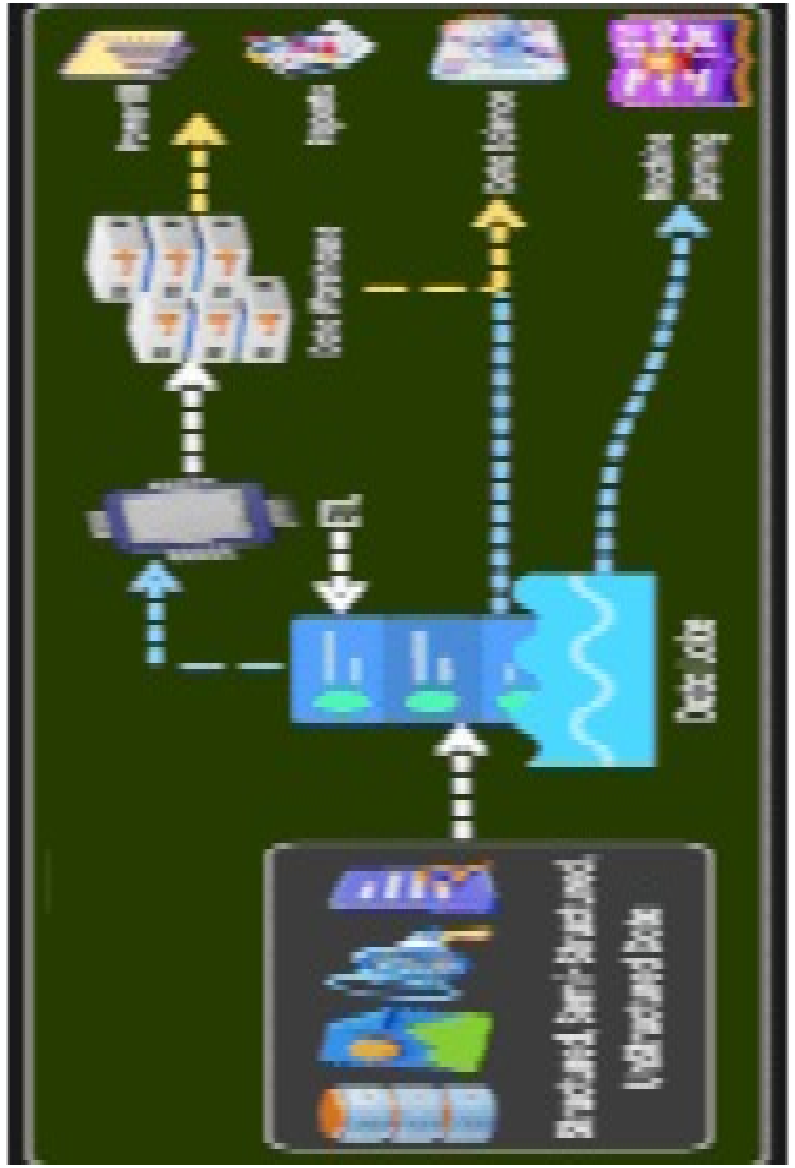


- You are running an ELT process
- The data is arriving at a very high velocity
- Incoming data is not structured and transformations should be done according to the format of the data
- Which one is the right storage for this?

- Extremely limitless and cheap storage costs
- Allows immerse amount of data of any size and type to be stored
- When needs to be queried or transformed, have access to nearly unlimited computing power by spinning cluster on demand with favourite tolls



Data Lake



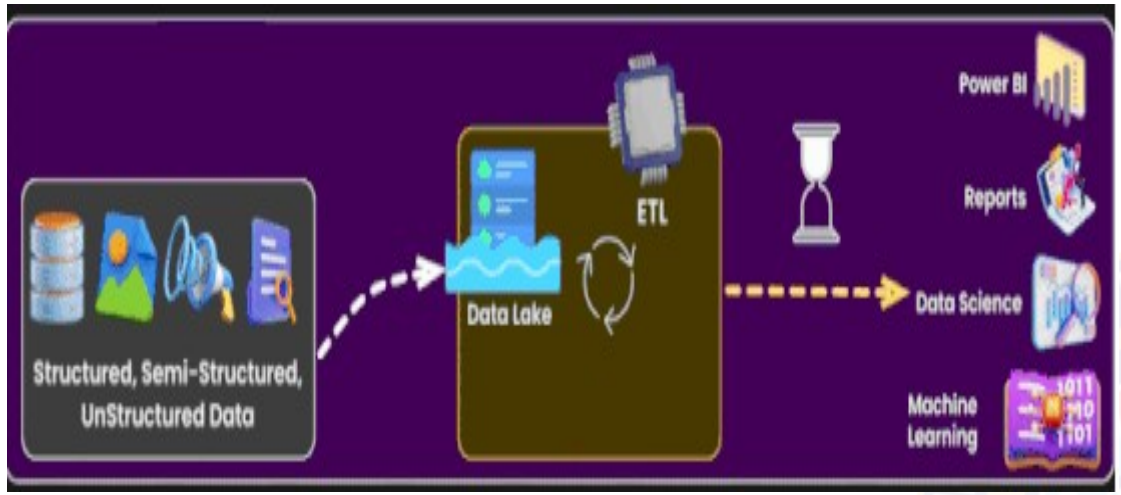
Drawbacks:

- Dumping ground
- Data swamps - **a disorganized and unstructured data storage environment**
- Dark data - the data, organizations collect, process and store during regular business activities, but fail to use for other purposes
- WORN – Write Once Read Never
- Grows unmanageable

Usage:

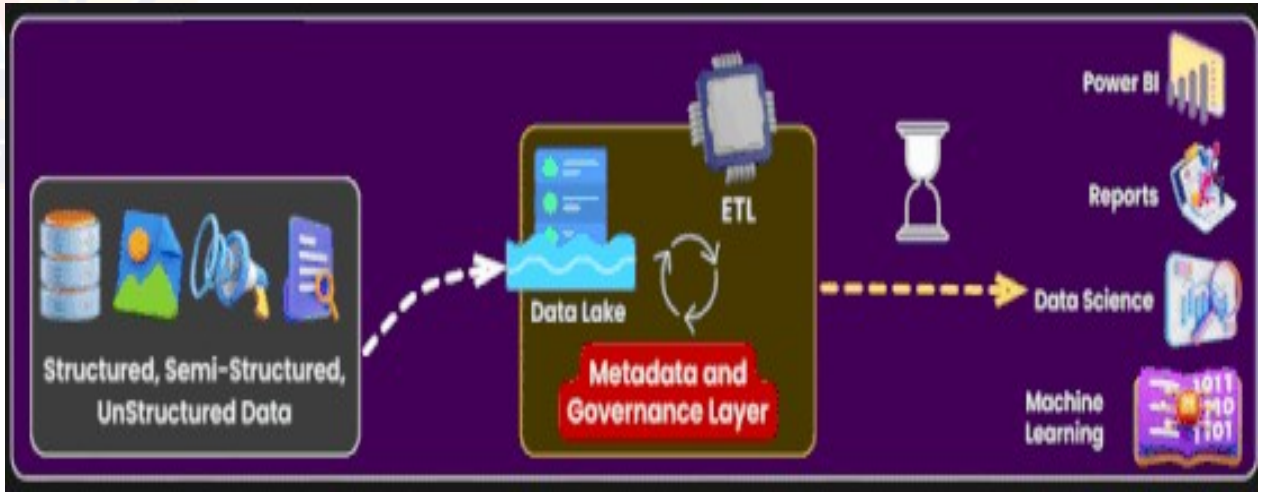
- Lot of companies uses datalakes. Eg: Facebook, Netflix
- Still they have lot of data as WORN

Data Lakehouse



- Databricks introduced data Lakehouse – convergence between data warehouse and data lakes
- Incorporates the controls, data management and data structures found in data warehouse while still housing data in object storage and supporting variety of query and transformation engines
- Supports Atomicity, Consistency, Durability and Isolation (ACID)

- How about combining the benefits of data warehouse on top of data lakes?
- Bringing in the governance and controls of data warehouse
- Solution?



Categorization – By Governance

DataMesh

- Recent response to sprawling monolithic data platforms, such as centralized data lakes and warehouse and great divide of data – operational and analytical data
- Attempts to invert the challenges of centralized data architecture – by applying concepts of domain driven design on data architecture

- Per Zhamak Dehghani,
- In order to decentralize the monolithic data platforms, we need to reverse how we think about data, its locality and ownership.
- Instead of flowing the data from domains into a centrally owned data lake or platform, domain need to host and serve their domain datasets in easily consumable way.
- **Four key components**
  - Domain-oriented decentralized data ownership and architecture
  - Data as a product
  - Self-serve data infrastructure as a platform
  - Federated computational governance.

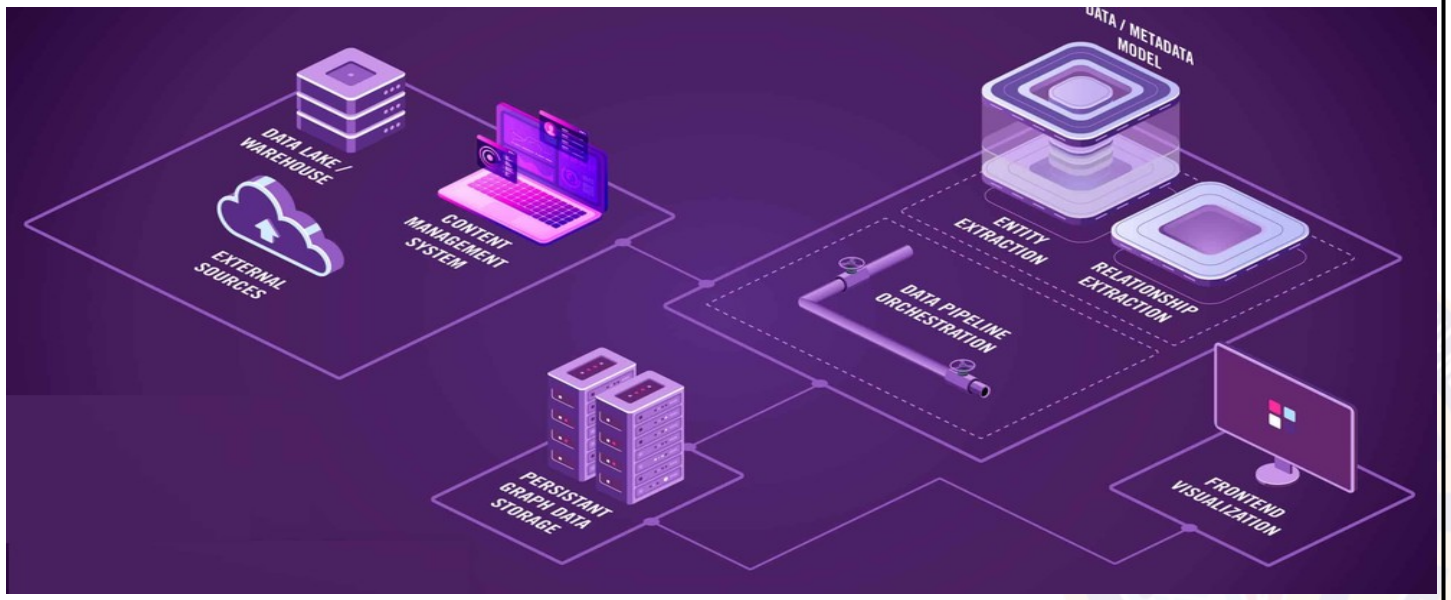
DataMesh

- The Data Mesh approach emphasizes domain-specific pipelines where each business area ingests, processes, and publishes data as autonomous data products.

- Self-serve platforms provide centralized infrastructure tools like data catalogs and monitoring capabilities, enabling domain teams to focus on domain-specific logic rather than infrastructure management.

- For example, an e-commerce team might expose order events through a curated dataset with comprehensive metadata describing schema, SLAs, and usage guidelines.

DataFabri
c



Key characteristics:

- Unified data access
- Seamless integration & orchestration
- Security, governance & compliance
- Scalability & flexibility
- Real-time insights and multi-cloud support

- Data fabric architectures create unified data layers that integrate information from lakes, warehouses, databases, and SaaS tools through APIs, change data capture, and virtualization technologies.

- A practical example would be creating a 360-degree customer view that combines sales data from Salesforce, transaction logs from a data lake, and social media sentiment via APIs.

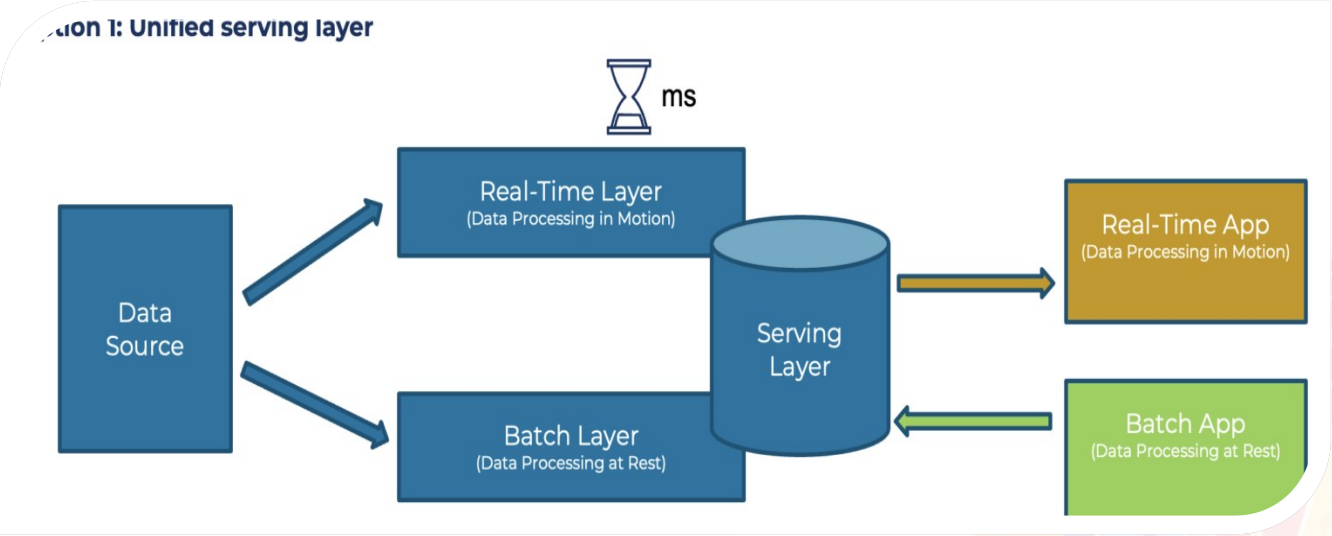
Categorization – By Processing

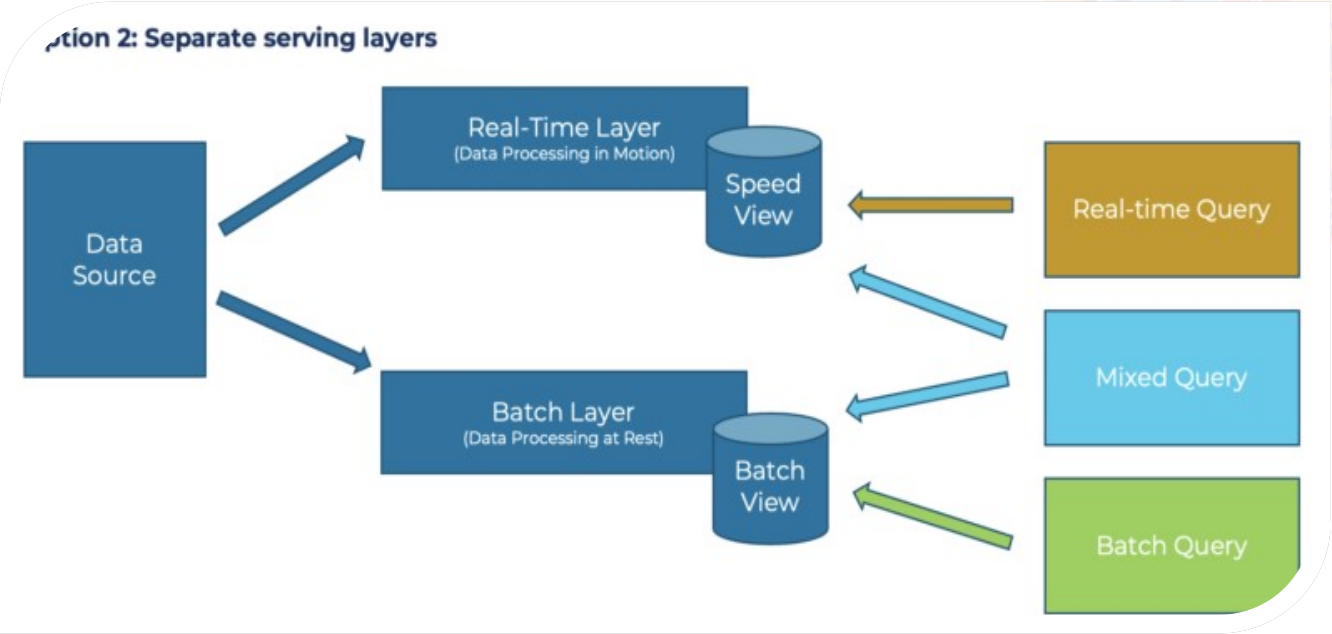
Lambda Architecture

- Consider a scenario
- A banking institute is building a fraud analysis and alert system. For each transaction, they want to store data for analysis & reporting and want to have a real-time engine to alert on possible fraud data coming in
- Let us deep dive
- They want to analyze the data and identify the fraud patterns and generate reports – this is an activity done on batch mode
- Alerting the backend staff on fraudulent transaction – this is a real time alert.
- So we need same data flowing for batch as well as stream processing
- Need to figure out how to reconcile batch and streaming data in single architecture

Solution: Lambda Architecture

Lambda Architecture

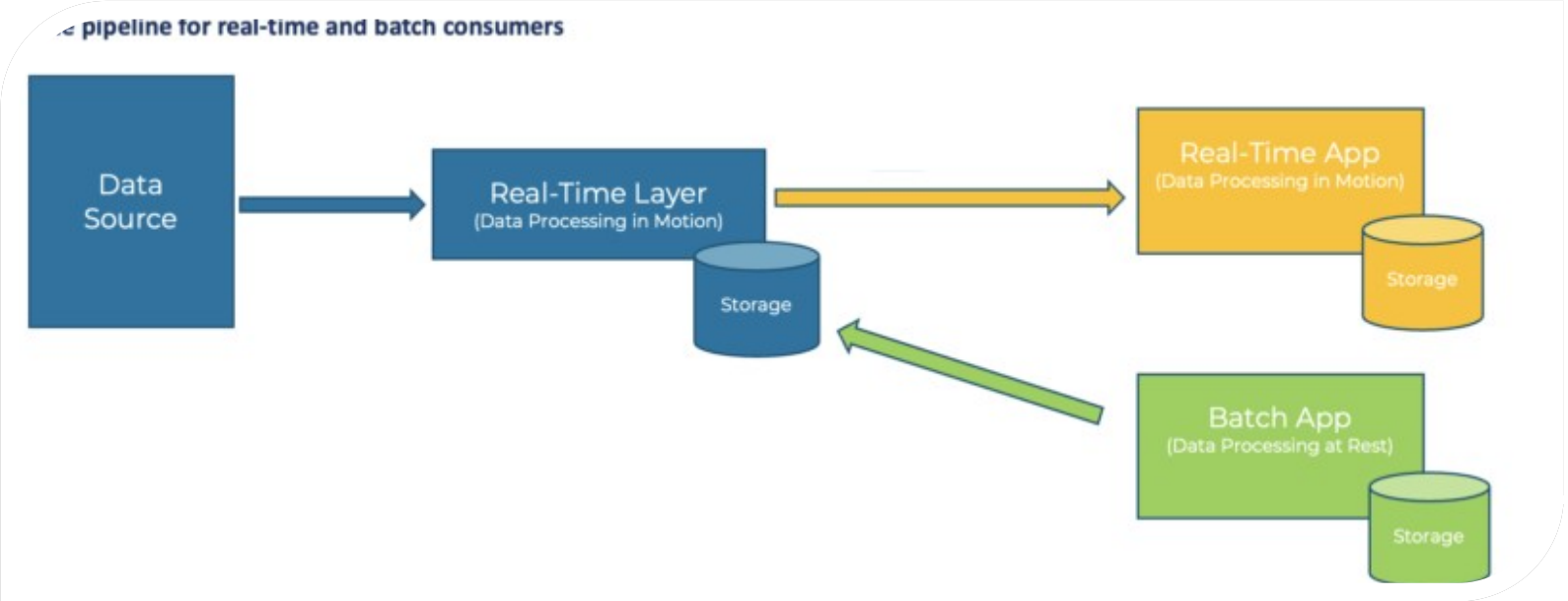




- Three systems : batch, streaming and serving are operating independently Source is immutable and append-only, sending data to two destinations : Stream and Batch
- In-stream processing serves data with lowest possible latency in speed layer, usually NoSQL DB
- Batch layer process and transforms data in data warehouse, creating precomputed and aggregated views of data
- Serving layer provides a combined view by aggregating query results from two layers(option 1) or separate service layers(option 2)
- **Challenge:** Managing multiple systems with different codebases is difficult, creating error-prone systems with code and data that are extremely difficult to reconcile!

Kappa Architecture

- As a response to shortcomings of Lambda architecture, Jay Kreps proposed an alternative
- Just use stream processing platform as backbone for all data handling
- Ingestion, storage and serving – facilitating a true event based architecture
- Real-time and batch processing can be applied seamlessly to same data
- By reading live event stream directly and replaying large chucks of data for batch processing



Challenges:

- Streaming itself is still a bit of mystery to many companies – easy to talk, harder to execute
- Turns out to be complicated and expensive in practice
- Batch storage and processing remain much more efficient and cost-effective for enormous historical data

Generalized Architecture of Big Data Systems

Introduction

Per https://www.forbes.com/sites/sap/2015/02/19/how-big-data-keeps-planes-in-the-air/

> Data collected by sensors on one aircraft covers more than 300,000 parameters with engine data one of the most important data points captured. The average commercial aircraft - like the Boeing 737 - creates 20 terabytes of engine information per hour. Multiply by an average six-hour cross-country flight from New York to Los Angles and you're now looking at 240 terabytes of data for every engine hour.

What is Big Data?

- Big data refers to massive, complex data sets that traditional data management systems cannot handle.
- When properly collected, managed and analyzed, big data can help organizations discover new insights and make better business decisions.
- Businesses collect large amounts of data—often measured in terabytes or petabytes—on everything from customer transactions and social media impressions to internal processes and

Traditional Data vs Big Data

- Primarily consists of structured data stored in relational databases. These databases organize data into clearly defined tables, making it easy to query using standard tools like SQL
- Traditional data analytics typically involves statistical methods and is well-suited for datasets with predictable formats and relatively small sizes.

- Encompasses massive datasets in various formats, including structured, semi-structured and unstructured data
.
- Complexity demands advanced analytical approaches—such as machine learning, data mining and data visualization—to extract meaningful insights.
- The sheer volume of big data also requires distributed processing systems to handle the data efficiently at scale.

5 Vs of Big Data

Volume

- The massive amount of data generated today—from web apps, Internet of Things (IoT) devices, transaction records and more—can be hard for any organization to manage.
- Traditional data storage and processing systems often struggle to handle it at scale.

Velocity

- The speed at which data flows into a system and big data moves quickly.
- Data arrives faster than ever, from real-time social media updates to high-frequency stock trading records.

Variety

- Refers to the many different formats that big data can take.
- Along with traditional structured data, big data can include unstructured data, such as free-form text, images and videos.

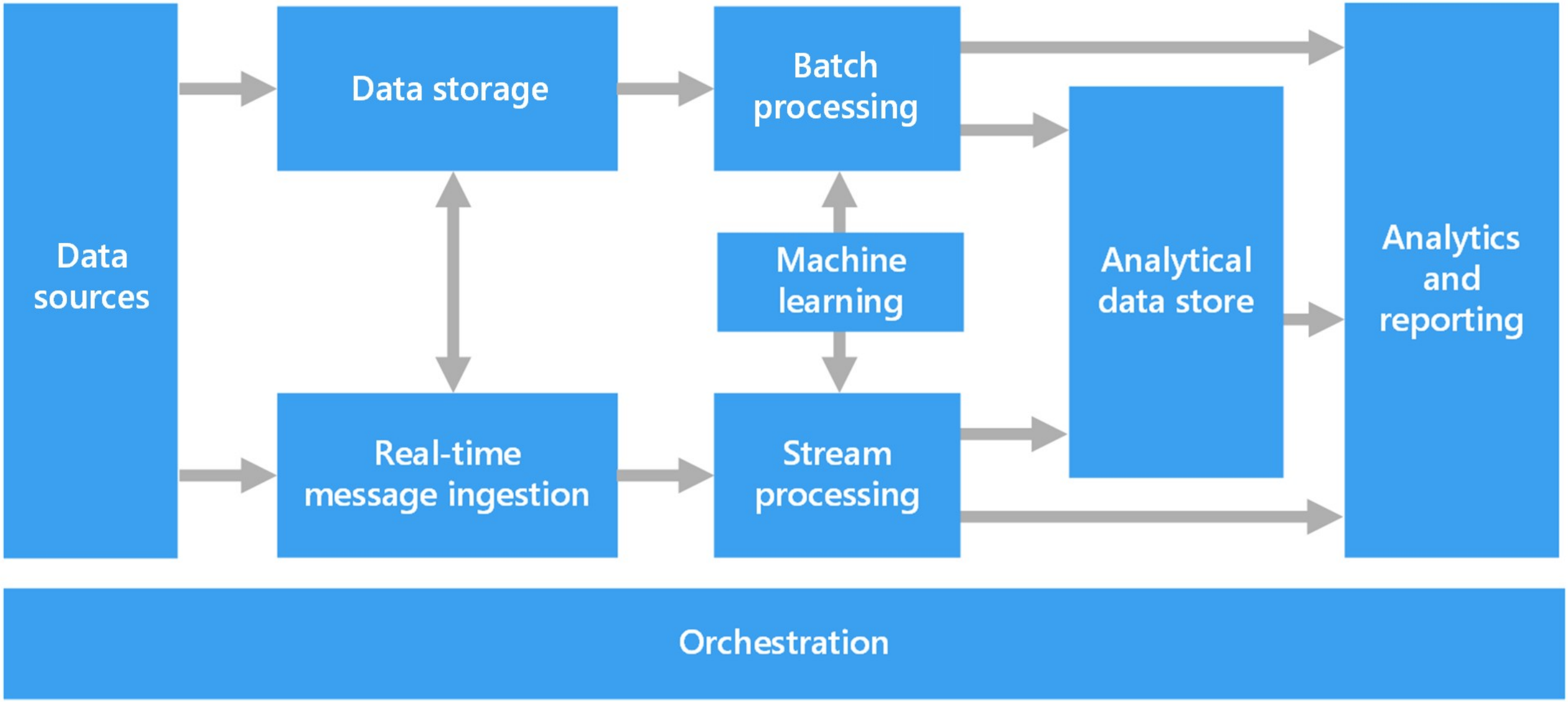
Veracity

- Refers to the accuracy and reliability of data.
- Because big data comes in such great quantities and from various sources, it can contain noise or errors, which can lead to poor decision-making.

Value

- Refers to the real-world benefits organizations can get from big data.
- These benefits include everything from optimizing business operations to identifying new marketing opportunities.

Components of Big Data Architecture



Big Data Architecture

- Manages the ingestion, processing, and analysis of data that's too large or complex for traditional database systems.
- The threshold for entering the realm of big data varies among organizations, depending on their tools and user capabilities.
- Some data arrives at a rapid pace and requires continuous collection and observation.
- Some data arrives more slowly, but in large chunks, and often in the form of decades of historical data.

Typical big data workloads:

- Batch processing of big data sources at rest
- Real-time processing of big data in motion
- Interactive exploration of big data
- Predictive analytics and machine learning

When to go for Big data architecture:

- Store and process data in volumes that are too large for a traditional database
- Transform unstructured data for analysis and reporting
- Capture, process, and analyze unbounded streams of data in real time or with low latency

Components of Big Data Architecture

- **Data Sources**

  All big data solutions start with one or more data sources like databases, files, IoT devices etc

- **Data Storage**

  Data for batch processing operations is typically stored in a distributed file store that can hold high volumes of large files in various formats.

- **Batch Processing**

  Because the data sets are so large, often a big data solution must process data files using long-running batch jobs to filter, aggregate, and otherwise prepare the data for analysis. Usually these jobs involve reading source files, processing them, and writing the output to new files.

- **Real-Time Message Ingestion**

  If the solution includes real-time sources, the architecture must include a way to capture and store real-time messages for stream processing.

Components of Big Data Architecture

- **Stream processing**

  After capturing real-time messages, the solution must process them by filtering, aggregating, and otherwise preparing the data for analysis. The processed stream data is then written to an output sink.

- **Analytical data store**

  Many big data solutions prepare data for analysis and then serve the processed data in a structured format that can be queried using analytical tools. The analytical data store used to serve these queries can be a Kimball-style relational data warehouse, as seen in most traditional business intelligence (BI) solutions.

- **Analysis and reporting**

  The goal of most big data solutions is to provide insights into the data through analysis and reporting.

- **Orchestration**

  Most big data solutions consist of repeated data processing operations, encapsulated in workflows, that transform source data, move data between multiple sources and sinks, load the

Big Data Architecture Benefits

- **Technology choices**

  Variety of technology options in open source and from vendors are available

- **Performance through parallelism**

  Big data solutions take advantage of parallelism, enabling high-performance solutions that scale to large volumes of data.

- **Elastic scale**

  All of the components in the big data architecture support scale-out provisioning, so that you can adjust your solution to small or large workloads, and pay only for the resources that you use.

- **Interoperability with existing solutions**

  The components of the big data architecture are also used for IoT processing and enterprise BI solutions, enabling you to create an integrated solution across data workloads.

Big Data Architecture Challenges

- **Complexity**

  Big data solutions can be extremely complex, with numerous components to handle data ingestion from multiple data sources. It can be challenging to build, test, and troubleshoot big data processes.

- **Skillset**

  Many big data technologies are highly specialized, and use frameworks and languages that are not typical of more general application architectures. On the other hand, big data technologies are evolving new APIs that build on more established languages.

- **Technology maturity**

  Many of the technologies used in big data are evolving. While core Hadoop technologies such as Hive and Pig have stabilized, emerging technologies such as Spark introduce extensive changes and enhancements with each new release.

Lambda & Kappa for Big Data Systems

Big Data Processing

Challenge:

- Working with large datasets, can take a long time to run the type of queries that clients need

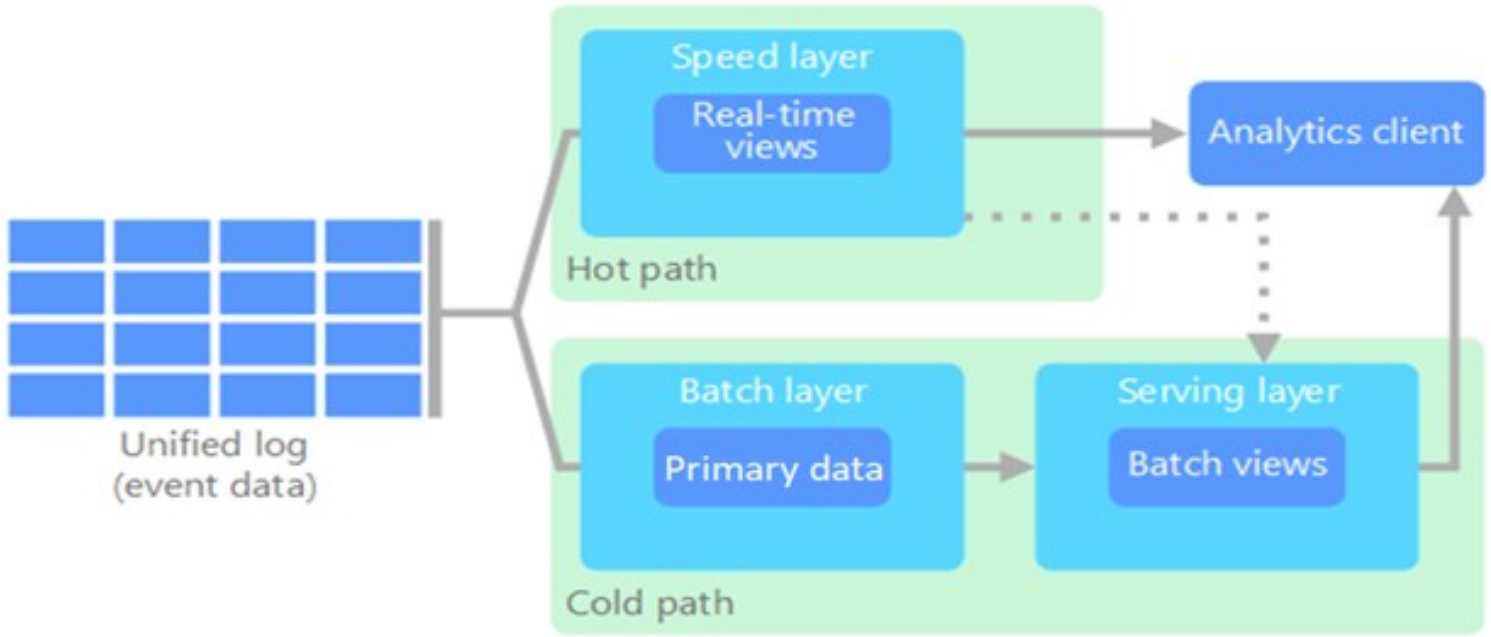
An Approach:

- These queries can't be performed in real time, and they often require distributed processing algorithms such as MapReduce that operate in parallel across the entire dataset.
- The query results are stored separately from the raw data and used for further querying.

Drawbacks of this approach:

- It introduces latency. If processing takes a few hours, a query might return results that are several hours old
- Ideally, should get some results in real time, potentially with a loss of accuracy, and combine these results with the results from batch analytics.

Lambda Architecture for Big Data Processing



Solution:

- The **Lambda architecture** addresses this problem by creating two paths for dataflow. All data that comes into the system goes through the following two paths:

  - A **batch layer** (cold path) stores all the incoming data in its raw form and performs batch processing on the data. The result of this processing is stored as a batch view.

  - A **speed layer** (hot path) analyzes data in real time. This layer is designed for low latency, at the expense of accuracy.

Lambda Architecture for Big Data Processing

Batch Layer(cold path)

- Feeds into a **serving layer** that indexes the batch view for efficient querying.
- Data that flows into the cold path isn't subject to the same low latency requirements.
- Provides high accuracy computation across large datasets but can take a long time.
- The client selects results from the cold path to display less timely but more accurate data.
- The raw data that's stored here is immutable.

Speed Layer(hot path)

- Updates the serving layer with incremental updates based on the most recent data.
- Data that flows into the hot path must be processed quickly because of latency requirements that the speed layer imposes.
- Quick processing ensures that data is ready for immediate use but can introduce inaccuracy.
- If the client needs to display timely, yet potentially less accurate data in real time, it acquires its result from the hot path.
- Changes to the value of a particular datum are stored as a new time-stamped event record.

Lambda Architecture for ML Workflows

- Lambda architectures support machine learning workloads by providing both historical data for model training and real-time data for inference.
- The batch layer enables training on comprehensive historical datasets using Fabric Data Science workloads.
- The speed layer facilitates real-time model inference and scoring.
- This dual approach allows for models trained on complete historical data while providing immediate predictions on incoming data streams.

Kappa Architecture for Big Data Processing

- It has the same basic goals as the Lambda architecture, but all data flows through a single path via a stream processing system.
- Similar to the Lambda architecture's batch layer, the event data is immutable and all of it is collected, instead of a subset of data.
- The data is ingested as a stream of events into a distributed, fault-tolerant unified log.
- These events are ordered, and the current state of an event is changed only by a new event being appended.
- Similar to the Lambda architecture's speed layer, all event processing is performed on the input stream and persisted as a real-time view.
- If you need to recompute the entire dataset (equivalent to what the batch layer does in the Lambda architecture), you can replay the stream.
- This process typically uses parallelism to complete the computation in a timely fashion.

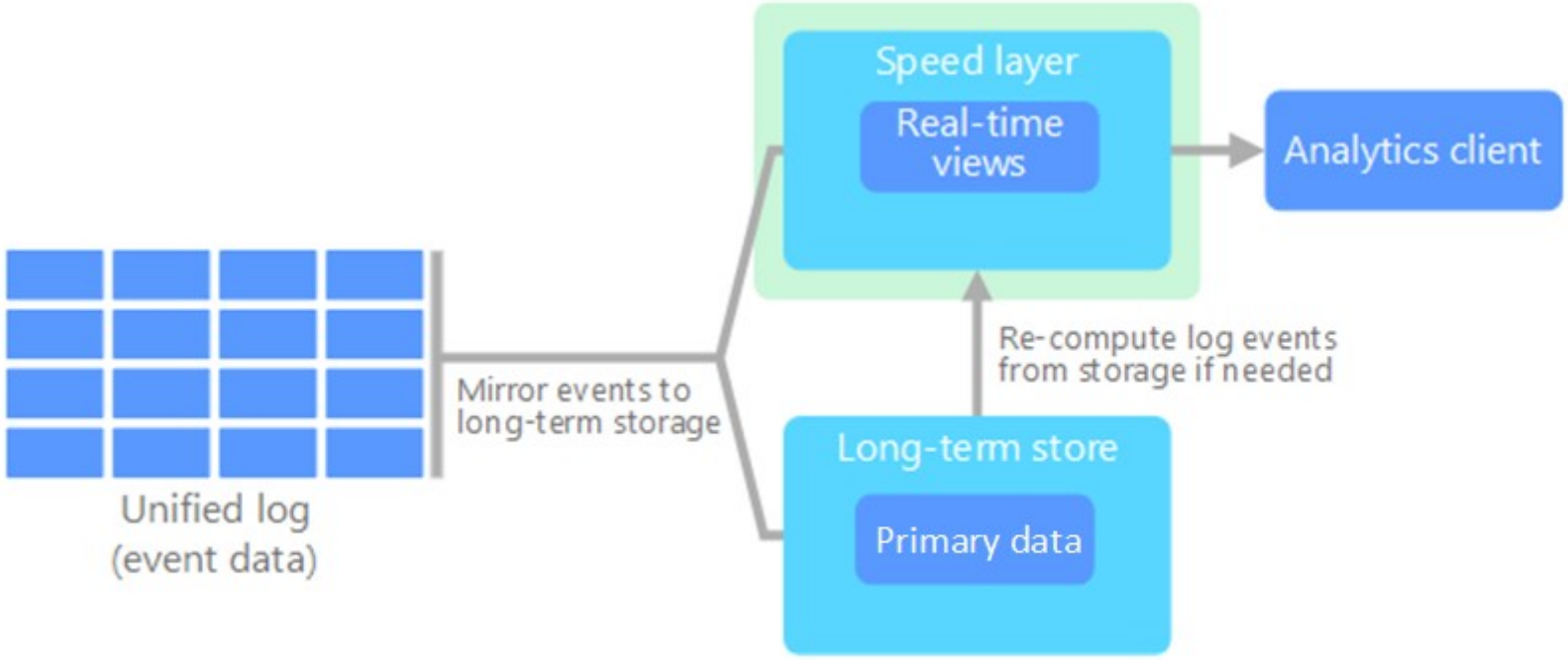
Kappa Architecture for Big Data Processing

Drawbacks of Lambda Architecture:

- Complex
- Processing logic appears in two different places, the cold and hot paths, via different frameworks.
- This process leads to duplicate computation logic and complex management of the architecture for both paths.

Solution:

Kappa Architecture



Kappa Architecture for ML Workflows

- Kappa architectures enable unified machine learning workflows by processing all data through a single streaming pipeline.
- This approach simplifies model deployment and maintenance since the same processing logic applies to both historical and real-time data.
- Can use Fabric Data Science workloads to build models that process streaming data, enabling continuous learning and real-time adaptation.
- The architecture supports online learning algorithms that update models incrementally as new data arrives.

References:

- Fundamentals of Data Engineering by Joe Reis and Matt Housley
- The Data Warehouse Toolkit by Ralph Kimball
- https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-forward/revisiting-data-architecture-for-next-gen-data-products
- https://www.researchgate.net/figure/Schematic-drawings-of-centralised-decentralised-and-hybrid-data-pooling-models-A_fig3_269170886
- https://www.linkedin.com/pulse/whats-data-lake-warehouse-house-mesh-right-you-your-rohit-yadav/
- https://www.kai-waehner.de/blog/2021/09/23/real-time-kappa-architecture-mainstream-replacing-batch-lambda/
- https://www.ibm.com/think/topics/big-data
- https://learn.microsoft.com/en-us/azure/architecture/databases/guide/big-data-architectures

Thank You!